Refine Search

Search Results -

Term	Documents
"6633916"	1
6633916S	0
"6633916".PN..USPT.	1
(6633916.PN.).USPT.	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L30

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, January 23, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L30	6633916.pn.	1	L30
L29	L28 and l13	1	L29
L28	64 bit41 or 64-bit$1	6690	L28
L27	l13 and l6	1	L27
L26	L25 and l13	1	L26
L25	translat$	241110	L25
L24	L23 and l4	1	L24
L23	l13 and (l8 and l7)	1	L23
L22	l13 and l4	1	L22
L21	L20 and l13	1	L21
L20	kernel	16475	L20

h e b b cg b e e ch

L19	L18 and l17	1	L19
L18	instruction$1	200650	L18
L17	L16 and l15	1	L17
L16	address$	353217	L16
L15	L14 and l13	1	L15
L14	cache or inhibit$	409907	L14
L13	5319760.pn.	1	L13
DB=TDBD; PLUR=YES; OP=ADJ			
L12	cache or inhibit$	3488	L12
DB=USPT; PLUR=YES; OP=ADJ			
L11	5319760.pn	0	L11
L10	L9 and firmware and kernel and l4	1	L10
L9	L8 and l7 and l6	156	L9
L8	64 bit$1 or 64-bit$1	13270	L8
L7	32 bit or 32-bit	27342	L7
L6	zero extended	338	L6
L5	l3 and l4	1	L5
L4	virtual$	247654	L4
L3	L2 and l1	1	L3
L2	bit$1	312953	L2
L1	6633916.pn.	1	L1

END OF SEARCH HISTORY

First Hit Fwd Refs

End of Result Set

☐ Generate Collection Print

L11: Entry 1 of 1 File: USPT Jun 7, 1994

DOCUMENT-IDENTIFIER: US 5319760 A
TITLE: Translation buffer for virtual machines with address space match

Abstract Text (1):
A central processing unit (CPU) executing a virtual memory management system employs a translation buffer for caching recently used page table entries. When more than one process is executing on the CPU, the translation buffer is usually flushed when a context switch is made, even though some of the entries would still be valid for commonly-referenced memory areas. An address space number feature is employed to allow entries to remain in the translation buffer for processes not currently executing, and the separate processes or the operating system can reuse entries in the translation buffer for such pages of memory that are commonly referenced. To allow this, an "address space match" entry in the page table entry signals that the translation buffer content can be used when the address tag matches, even though the address space numbers do not necessarily match. When executing virtual machines on this CPU, with a virtual machine monitor, the address space match feature is employed among processes of a virtual machine, but an additional entry is provided to disable the address space match feature for all address space numbers for the virtual machine monitor.

Brief Summary Text (5):
This invention relates to digital computers, and more particularly to a virtual memory management system for a central processing unit (CPU) executing multiple processes.

Brief Summary Text (6):
In the above-identified copending application Ser. No. 547,630, a reduced instruction set processor chip is shown which implements a virtual memory management system. A virtual address is translated to a physical address in such a system before a memory reference is executed, where the physical address is that used to access the main memory. The physical addresses are maintained in a page table, indexed by the virtual address, so whenever a virtual address is presented the memory management system finds the physical address by referencing the page table. At a given time, a process executing on such a machine will probably be using only a few pages, and so these most-likely used page table entries are kept in a translation buffer within the CPU chip itself, eliminating the need to make a memory reference to fetch the page table entry.

Brief Summary Text (8):
A number of processes (tasks) may be executing in a time-shared manner on a CPU at a given time, and these processes will each have their own areas of virtual memory. The operating system itself will contain a number of pages which must be referenced by each one of these processes. The pages thus shared by processes will thus best be kept in main memory rather than swapped out, and also the page table entries for such pages will preferably remain in the translation buffer since continuing reference will be made to these. However, the translation buffer is usually flushed by invalidating all entries when a context switch is made.

Brief Summary Text (10):
In operating a CPU using virtual machines, each virtual machine functions as if it were an independent processor, and each virtual machine has a virtual machine operating system and a number of processes running, just as when only one CPU is functioning. Virtual machines are described by Siewiorek et al in "Computer Structures: Principles and Examples" published by McGraw-Hill, 1982, pp. 227-228. To operate these virtual machines, a virtual machine monitor (another execution level) is implemented. As before, there are pages used by the operating system that are used by all of the processes on a virtual machine. Again, performance is improved if entries for the pages remain in the translation buffer when making a context switch between processes or between virtual machines.

Brief Summary Text (11):
The several virtual machines and the virtual machine monitor running on a CPU must have their memory spaces kept separate and isolated from one another, but yet maximize system performance. To this end, the virtual machines and the virtual machine monitor must be able to use the same virtual addresses for different purposes. However, when context switching from one virtual machine to another, or to or from the virtual machine monitor, needlessly flushing entries in the translation buffer which will be used in the new context imposes a performance penalty. Therefore it is important to offer both address space numbers and the match feature when implementing virtual machines.

Brief Summary Text (13):
In accordance with one embodiment of the invention, a CPU executing a virtual memory management system employs an address space number feature to allow entries to remain in the translation buffer for processes not currently executing, and the separate processes or the operating system can reuse entries in the translation buffer for such pages of memory that are commonly referenced. To allow this, an "address space match" entry in the page table entry signals that the translation buffer content can be used when the address tag matches, even though the address space numbers do not necessarily match. When executing virtual machines on this CPU, with a virtual machine monitor, the address space match feature is employed within a virtual machine, but an additional entry is provided to disable the address space match feature for all address space numbers for the virtual machine monitor. In another embodiment, an additional entry is provided in the translation buffer to restrict the address space match feature to those address spaces associated with a single virtual machine or virtual machine monitor.

Drawing Description Text (10):
FIG. 8 is a diagram of the format of a virtual address used in the CPU of FIGS. 1-5;

Drawing Description Text (13):
FIG. 11 is a diagram of the virtual-to-physical address mapping used in the system of FIGS. 1-5;

Drawing Description Text (15):
FIG. 13 is a table of address space numbers for an example of operating the system of FIGS. 1-5 with virtual machines;

Detailed Description Text (3):
The CPU 10 is of a single-chip integrated circuit device, in an example embodiment, although features of the invention could be employed as well in a processor constructed of integrated circuit devices mounted on boards. Within the single chip an integer execution unit 16 (referred to as the "E-box") is included, along with a floating point execution unit 17 (referred to as the "F-box"). Instruction fetch and decoding is performed in an instruction unit 18 or "I-box". An address unit or "A-box" 19 performs the functions of address generation, memory management, write buffering and bus interface; the virtual address system with translation buffer

according to the invention is implemented in the address unit 19. The memory is hierarchical, with on-chip instruction and data caches being included in the instruction unit 18 and address unit 19 in one embodiment, while a larger, second-level cache 20 is provided off-chip, being controlled by a cache controller in the address unit 19.

Detailed Description Text (6):
The instruction unit 18 contains address generation circuitry 29, including a branch prediction circuit 30 responsive to the instructions in the instruction stream to be loaded into register 22. The prediction circuit 30 is used to predict branch addresses and to cause address generating circuitry 29 to prefetch the instruction stream before needed. The virtual PC (program counter) 33 is included in the address generation circuitry 29 to produce addresses for instruction stream data in the selected order.

Detailed Description Text (7):
The instruction unit 18 contains a fully associative translation buffer (TB) 36 to cache recently used instruction-stream address translations and protection information for 8 Kbyte pages. Although 64-bit addresses are nominally possible, as a practical matter 43-bit addresses are adequate. Every cycle the 43-bit virtual program counter 33 is presented to the instruction stream TB 36. If the page table entry (PTE) associated with the virtual address from the virtual PC is cached in the TB 36 then the page frame number (PFN) and protection bits for the page which contains the virtual PC are used by the instruction unit 18 to complete the address translation and access checks. A physical address is thus applied to the address input 37 of the instruction cache 21, or if there is a cache miss then this instruction stream physical address is applied by the bus 38 through the address unit 19 to the cache 20 or memory 12.

Detailed Description Text (10):
The datapath translation buffer 48 caches a number (e.g., thirty-two) of the recently-used data-stream page table entries (as described below) for pages of 8 Kbyte size. Each entry supports any of four granularity hint block sizes, and a detector 55 is responsive to the granularity hint as described in application Ser. No. 547,630 to change the number of low-order bits of the virtual address passed through from virtual address bus 56 to the physical address bus 54.

Detailed Description Text (11):
For load and store instructions, the effective 43-bit virtual address is presented to TB 48 via bus 56. If the PTE of the supplied virtual address is cached in the TB 48, the PFN and protection bits for the page which contains the address are used by the address unit 19 to complete the address translation and access checks.

Detailed Description Text (14):
The instruction cache 21 may be 8 Kbytes, or 16 Kbytes, for example, or may be larger or smaller, depending upon die area. Although described above as using physical addressing with a TB 36, it may also be a virtual cache, in which case it will contain no provision for maintaining its coherence with memory 12. If the cache 21 is a physical addressed cache the chip will contain circuitry for maintaining its coherence with memory: (1) when the write buffer 50 entries are sent to the bus interface 52, the address will be compared against a duplicate instruction cache 21 tag, and the corresponding block of instruction cache 21 will be conditionally invalidated; (2) the invalidate bus will be connected to the instruction cache 21.

Detailed Description Text (20):
One type is a memory (i.e., load and store) instruction 70, which contains a 6-bit opcode in bits <31:26>, two 5-bit register address fields Ra and Rb in bits <25:21> and <20:16>, and a 16-bit signed displacement in bits <15:0>. This instruction is used to transfer data between registers 43 and memory (memory 12 or caches 59 or

20), to load an effective address to a register of the register file 43, and for subroutine jumps. The displacement field <15:0> is a byte offset; it is sign-extended and added to the contents of register Rb to form a virtual address. The virtual address is used as a memory load/store address or a result value depending upon the specific instruction.

Detailed Description Text (21):
The branch instruction format 71 is also shown in FIG. 7, and includes a 6-bit opcode in bits <31:26>, a 5-bit address field in bits <25:21>, and a 21-bit signed branch displacement in bits <20:0>. The displacement is treated as a longword offset, meaning that it is shifted left two bits (to address a longword boundary), sign-extended to 64-bits and added to the updated contents of PC 33 to form the target virtual address (overflow is ignored).

Detailed Description Text (26):
Referring to FIG. 8, the format 76 of the virtual address asserted on the internal address bus 56 is shown. This address is nominally 64-bits in width, but of course practical implementations at present use much smaller addresses. For example, an address of 43-bits provides an addressing range of 8-Terabytes. The format includes a byte offset 77 of, for example, 13-bits to 16-bits in size, depending upon the page size employed. If pages are 8-Kbytes, the byte-within-page field 77 is 13-bits, while for 16-Kbyte pages the field 77 is 14-bits. The format 76 as shown includes three segment fields 78, 79 and 80, labelled Seg1, Seg2 and Seg3, also of variable size depending upon the implementation. The segments Seg1, Seg2, and Seg3 can be 10-to-13 bits, for example. If each segment size is 10-bits, then a segment defined by Seg3 is 1K pages in length, a segment for Seg2 is 1M pages, and a segment for Seg1 is 1 G pages. The page frame number (PFN) field in the PTE is always 32-bits wide; thus, as the page size grows the virtual and physical address size also grows.

Detailed Description Text (29):
A page table entry or PTE 81, as stored in the translation buffers 36 or 48 or in the page tables set up in the memory 12 by the operating system, is illustrated in FIG. 9. The PTE 81 is a quadword (64-bits) in width, and includes a 32-bit page frame number or PFN 82 at bits <63:32>, as well as certain software and hardware control information in a field 83 having bits <15:0> as set forth in Table A to implement the protection features and the like.

Detailed Description Text (30):
The translation buffers 36 and 48 store a number of the page table entries 81, each associated with a tag consisting of certain high-order bits of the virtual address to which this PFN is assigned by the operating system. For example, the tag may consist of the fields 78, 79 and 80 for the Seg1, Seg2 and Seg3 values of the virtual address 76 of FIG. 8. In addition, each entry contains a valid bit, indicating whether or not the entry has been invalidated, as when the TB is flushed. It is conventional to flush the TB when a context switch is made, invalidating all the entries; the features of the invention, however, allow continued use of entries still useful, so performance is improved. To this end, the translation buffers 36 and 48 include in addition an address space number field, perhaps sixteen bits in width, loaded from the process control block as will be described.

Detailed Description Text (31):
Referring to FIG. 10, the virtual address 76 on the bus 56 (seen in FIG. 8) is used to search for tag match for a PTE in the translation buffer, and, if not found, then Seg1 field 78 is used to index into a first page table 85 found at a base address stored in an internal register 86 referred to as the page table base register. The entry 87 found at the Seg1 index in table 85 is the base address for a second page table 88, for which the Seg2 field 79 is used to index to an entry 89. The entry 89 points to the base of a third page table 90, and Seg3 field 80 is

used to index to a PTE 91. The physical page frame number from PTE 91 is combined with the byte offset 77 from the virtual address, in adder 92, to produce the physical address on bus 54. As mentioned above, the size of the page mapped by a PTE, along with size of the byte offset 77, can vary depending upon the granularity hint.

Detailed Description Text (32):
According to the invention, in addition to matching the tag field of the virtual address 76 with the tag field 93 of the translation buffer 36 or 48, an address space number 94 stored in the translation buffer is compared to the address space number in the current state of the CPU 10, stored in an ASN register 95 which is one of the internal processor registers. If the address space match field (bit <4>, Table A) is clear (zero), the current ASN and the field 94 must match for the PTE to be used, but if set (logic one) then there need not be a match, i.e., the address space numbers are ignored. A particular feature of the invention, however, is an additional match-disable bit 96 stored for each PTE, disabling the address space match feature under certain conditions (i.e., when the virtual machine monitor process is being executed).

Detailed Description Text (33):
The match-disable bit is not required to be maintained on each entry in the translation buffer. Whether or not address-space matches should be disabled is properly a function of the execution environment of the CPU rather than of the virtual address. When the virtual machine monitor is being executed (as discussed below), address-space matches are disabled; when a virtual machine or some process in a VM is being executed, address-space matches are enabled. In another embodiment of the invention, the match-disable bit could be stored globally in the translation buffer. In yet another embodiment of the invention, the match-disable bit could be maintained in the CPU itself. In either case, its value would be changed on transition from the VMM to a VM or from a VM to the VMM, and its current value must be made available to the match comparison logic in the translation buffer.

Detailed Description Text (34):
The CPU 10 generates memory references by first forming a virtual address 76 on bus 56, representing the address within the entire virtual range 97 as seen in FIG. 11, defined by the 43-bit address width referred to, or that portion of the address width used by the operating system. Then using the page tables 85, 88, 90 in memory, or the translation buffer 36 or 48, the virtual address is translated to a physical address represented by an address map 98; the physical memory is constrained by the size of the main memory 12. The translation is done for each page (e.g., an 8 Kbyte block), so a virtual page address for a page 99 in the virtual memory map 97 is translated to a physical address 99' for a page (referred to as a page frame) in the physical memory map 98. The page tables are maintained in memory 12 or cache 20 to provide the translation between virtual address and physical address, and the translation buffer is included in the CPU to hold the most recently used translations so a reference to the page tables in memory 12 need not be made in most cases to obtain the translation before a data reference can be made; the time needed to make the reference to a page table in memory 12 would far exceed the time needed to obtain the translation from the translation buffer. Only the pages used by tasks currently executing (and the operating system itself) are likely to be in the physical memory 12 at a given time; a translation to an address 99' is in the page table 85, 88, 90 for only those pages actually present in physical memory 12. When the page being referenced by the CPU 10 is found not to be in the physical memory 12, a page fault is executed to initiate a swap operation in which a page from the physical memory 12 is swapped with the desired page maintained in the disk memory 13, this swap being under control of the operating system. Some pages in physical memory 12 used by the operating system kernel, for example, or the page tables 85, 88, 90 themselves, are in fixed positions and may not be swapped to disk 13 or moved to other page translations; most pages used by executing tasks, however, may be moved freely within the physical memory 12 by

merely keeping the page tables updated.

Detailed Description Text (35):
A process (or task) is a basic entity that is scheduled for execution by the CPU 10. A process represents a single thread of execution and consists of an address space and both hardware and software context. The hardware context is defined by the integer registers 43 and floating point registers 61, the processor status contained in internal processor registers, the program counter 33, four stack pointers, the page table base register 86, the address space number 95, and other values depending upon the CPU design. The software context of a process is defined by operating system software and is system dependent. A process may share the same address space with other processes or have an address space of its own; there is, however, no separate address space for system software, and therefore the operating system must be mapped into the address space of each process. In order for a process to execute, its hardware context must be loaded into the integer registers 43, floating point registers 61, and internal processor registers. While a process is executing, its hardware context is continuously updated, as the various registers are loaded and written over. When a process is not being executed, its hardware context is stored in memory 12. Saving the hardware context of the current process in memory, followed by loading the hardware context for a new process, is referred to as context switching. Context switching occurs as one process after another is scheduled by the operating system for execution. The hardware context of a process is defined by a privileged part and nonprivileged part. The privileged part is stored in memory in a 128-byte block 100 as shown in FIG. 12 when a process is not executing, and context is switched by a privileged instruction. There is one block 100 for each process. The nonprivileged part is context switched by the operating system software.

Detailed Description Text (36):
Referring to FIG. 12, the context block 100 contains four stack pointers in fields 101, these being stack pointers for the kernel, the executive, the supervisor and the user. The page table base register 86 is in field 102. The address space number for this process (to be loaded to register 95) is in field 103. Other fields 104 are for values not material here. The location of this block 100 in memory is specified for the current process by a context block base register 105. A swap context instruction saves the privileged context of the current process into the context block specified by this register 105, loads a new value into the register 105, and then loads the privileged context of the new process from the new block 100 into the appropriate hardware registers 43, etc.

Detailed Description Text (38):
In the page table entry 81 of FIG. 9 and Table A, there is a field (bit <4>) called "address space match." This feature allows an operating system to designate locations in the system's virtual address space 97 which are shared among all processes. Such a virtual address refers to the same physical address in each process's address space.

Detailed Description Text (39):
The CPU 10 of FIGS. 1-5 may employ a "virtual machine system" which uses a combination of hardware, firmware, and software mechanisms to create the illusion of multiple, independent simulated or virtual computers each running on the same physical computer. Virtual machine systems create a set of virtual machines (VMs) in an analogous fashion to how time sharing systems create a set of independent user processes. In virtualizing the architecture of FIG. 1-5, it is desirable from performance and functional standpoints to provide the address-space-match feature to virtual machines through hardware means.

Detailed Description Text (40):
Virtual machines are created by a layer executing on the CPU called the virtual machine monitor (VMM). The VMM is in control of the hardware, including memory

management (address translation and protection mechanisms) and the page tables. Each virtual machine runs in a separate virtual address space in the range 97, and has a distinct set of address space numbers assigned by the VMM. The VMM also runs in its own, independent set of virtual address spaces in the range 97.

Detailed Description Text (41):
The purpose of the invention is to maximize system performance, while allowing the virtual machines to run in kernel mode so they can execute privileged instructions (otherwise they would have to use traps for these operations, at considerable performance penalty); to do this the VMM must constrain the VMs. The problem addressed in implementing this invention is to keep the address spaces of the several VMs and the VMM itself separate from each other, while at the same time maximizing system performance by providing the match function in the TB to the VMs. A further constraint on the solution is that it is expected that the VMs and the VMM will use the same virtual addresses for different purposes. Thus, it is not a solution to allocate separate address regions to the VMM from those allocated to the VMs.

Detailed Description Text (42):
To describe this invention, an example is illustrative. Suppose that the hardware provides fifteen address spaces for use by software, numbered ASN-0 through ASN-14. Assume that address spaces ASN-0 and ASN-9 are dedicated for use by the VMM. On this example system, there are running two virtual machines, A and B, each of which is running five user processes. One possible assignment of address spaces to this mix would be to dedicate address spaces ASN-1 through ASN-5 to VM A and address spaces ASN-6, -7, -8, -10, and -11 to VM B. This example is illustrated in FIG. 13.

Detailed Description Text (43):
To preserve system security and integrity, it is required that the two virtual machines be completely independent. That means that they cannot reference each other's memory. Similarly, the VMM's memory must also be isolated from the virtual machines. However, within a virtual machine, the address-space-match feature should work correctly, allowing the individual processes to share memory, under control of each VM's operating system.

Detailed Description Text (45):
This kind of TB can be used in a virtual machine system at a considerable performance penalty. In order to enforce the memory isolation requirements, the TB must be completely flushed whenever (1) there are any entries in the TB that have the match field (bit <4>) indicating match all, and (2) any of the following events occurs: (a) the currently executing entity on the real machine changes from one VM to another VM, (b) the currently executing entity on the real machine changes from some VM to the VMM, or (c) the currently executing entity on the real machine changes from the VMM to any VM.

Detailed Description Text (46):
To minimize the TB flushes, and thus improve overall system performance, some restriction is imposed on the VMM and the TB construction is changed, according to the invention. The restriction imposed on the VMM is to require it to use the disable match feature. This reserves the address space match feature for use by the virtual machines, and guarantees that no VMM address will be mapped by the TB with the match field set.

Detailed Description Text (49):
where ADDR.TAG is the tag fields 78, 79, 80 of the virtual address 76, TB.TAG is the tag 93 in the translation buffer, CPU.ASN is the value stored in the processor register 95 from the field 103 of the control block, TB.ASN is the address space number stored in the field 96 of the translation buffer, TB.ASM is the match bit, and TB.DIS is the diable bit in field 96 of the translation buffer. These values

are applied to a logic circuit 106 in the A-box of FIG. 4 to generate a match signal to indicate whether or not the PTE selected in the TB is to be used.

Detailed Description Text (53):
In another embodiment of the invention, a multi-bit field for a virtual machine number VMN is added to each translation buffer entry as seen in FIG. 15, instead of the single-bit disable indicator as discussed above. Likewise, a multi-bit virtual machine number VMN is added to the state of the CPU in an internal processor register, like the address space number field 95. The translation buffer then contains logic to match a virtual address on the tags and match the ASNs, as before, and also logic to match on the virtual machine numbers. The logic implemented may be expressed

Detailed Description Text (54):
where CPU.VMN is the virtual machine number stored as machine state and TB.VMN is the VMN field in the translation buffer entry.

Detailed Description Text (55):
In this embodiment, each VM is assigned one (or more) VMN, and the VMM is assigned one (or more) VMN, and each must maintain operation using a VMN distinct to itself. In contrast to the previous embodiment, however, the translation buffer does not need to be cleared upon any switch between VMs. An additional advantage is the virtual machine monitor can use the address space match to share translation buffer entries among its processes. The disadvantage of this embodiment is that more bits are needed in the translation buffer entries.

Detailed Description Paragraph Table (4):
TABLE A _____ Page Table Entry Fields in the page table entry are interpreted as follows: Bits Description
_____ <0> Valid (V) - Indicates the validity of the PFN field. <1> Fault On Read (FOR) - When set, a Fault On Read exception occurs on an attempt to read any location in the page. <2> Fault On Write (FOW) - When set, a Fault On Write exception occurs on an attempt to write any location in the page. <3> Fault on Execute (FOE) - When set, a Fault On Execute exception occurs on an attempt to execute an instruction in the page. <4> Address Space Match (ASM) - When set, this PTE matches all Address Space Numbers. For a given VA, ASM must be set consistently in all processes. <6:5> Granularity hint (GH) - Software may set these bits to a non-zero value to supply a hint to the translation buffer that a block of pages can be treated as a single larger page. <7> Reserved for future use. <8> Kernel Read Enable (KRE) - This bit enables reads from kernel mode. If this bit is a 0 and a LOAD or instruction fetch is attempted while in kernel mode, an Access Violation occurs. This bit is valid even when V = 0. <9> Executive Read Enable (ERE) - This bit enables reads from executive mode. If this bit is a 0 and a LOAD or instruction fetch is attempted while in executive mode, an Access Violation occurs. This bit is valid even when V = 0. <10> Supervisor Read Enable (SRE) - This bit enables reads from supervisor mode. If this bit is a 0 and a LOAD or instruction fetch is attempted while in supervisor mode, an Access Violation occurs. This bit is valid even when V = 0. <11> User Read Enable (URE) - This bit enables reads from user mode. If this bit is a 0 and a LOAD or instruction fetch is attempted while in user mode, an Access Violation occurs. This bit is valid even when V = 0. <12> Kernel Write Enable (KWE) - This bit enables writes from kernel mode. If this bit is a 0 and a STORE is attempted while in kernel mode, an Access Violation occurs. This bit is valid even when V = 0. <13> Executive Write Enable (EWE) - The bit enables writes from executive mode. If this bit is a 0 and a STORE is attempted while in executive mode, an Access Violation occurs. <14> Supervisor Write Enable (SWE) - This bit enables writes from supervisor mode. If this bit is a 0 and a STORE is attempted while in executive mode, an Access Violation occurs. <15> User Write Enable (UWE) - This bit enables writes from user mode. If this bit is a 0 and a STORE is attempted while in user mode, Access Violation occurs. <31:16> Reserved for software. <63:32> Page Frame Number (PFN) - The PFN field

always points to a page boundary. If V is set, the PFN is concatenated with the Byte Within Page bits of the virtual address to obtain the physical address. If V is clear, this field may be used by software.

Other Reference Publication (3):
Hall et al., "Virtualizing the VAX Architecture", Proc. 18th Int'l. Symp. on Computer Architecture (Assoc. for Comp. Mach.) 1990.

CLAIMS:

1. A method of operating a processor having a translation buffer for translating a virtual address to a physical address, said method comprising the steps of:

storing in said translation buffer a plurality of page table entries, each page table entry containing a page frame number indexed by a virtual address tag;

also storing in said translation buffer for each said page table entry an address space number, and storing in said translation buffer for each said page table entry an address space match entry; where said address space number is a value corresponding to a process executed on said processor, said match entry is a field having one value indicating that the address space number is to be required to be matched and having another value indicating that the address space number is not required to be matched;

storing a current number in said processor as part of a state of said processor;

and storing a third match value having one condition indicating that said match entry is to be disabled and having another condition indicating that said match entry is not to be disabled;

comparing said virtual address tag with a field of a virtual address generated by said processor, and also comparing said address space number with said current number, if comparing said virtual address tag with said field of said virtual address produces a match, and

if said step of comparing of said address space number and said current number produces said match, and said match entry is of said one value, then using said page frame number for a memory reference; and

if said match entry is said another value, then using said page frame number for said memory reference regardless of whether said address space number matches said current number, if said third match value is in said one condition.

2. The method of operating said processor according to claim 1 wherein said processor is executing a plurality of virtual machines, each having a number of processes, and is executing a virtual machine monitor.

3. The method of operating said processor according to claim 2 wherein said third match value is in said one condition for all page table entries in said translation buffer for said virtual machine monitor.

8. A method of operating a processor system having a CPU and a memory, the CPU having a translation buffer for translating virtual addresses to physical addresses, said method of operating comprising the steps of:

storing in said translation buffer a number of page table entries, each page table entry containing a virtual address tag, a page frame number, and an address space number to characterize a location in said memory referenced by said page frame number;

also storing in said translation buffer for each page table entry (a) an address space match indication having one value indicating that said address space number must match a current value stored as part of a state of said processor and having another value indicating that said address space number need not match said current value, and (b) a match disable indication having a first value specifying that said address space match indicator is to be operable for said entry and having a second value indicating that said address space match indicator is not to be operable for an entry;

comparing a field of a virtual address generated by said processor with a virtual address tag of one of said page table entries in said translation buffer, and

if said step of comparing indicates a match, and said address space match indication is of said another value, and said address space number matches a value stored as part of the state of said processor, then using said page frame number for addressing said memory;

if said step of comparing indicates said match, and said address space match indication is of said one value, regardless of whether said address space number matches said current value, using said page frame number for addressing said memory;

if said step of comparing indicates a match, and said match disable indication is of said first value, regardless of whether said address space match indication is of said one value or said another value, then using said page frame number for addressing said memory only if said address space number matches said current value.

11. The method of operating said processor system according to claim 8 wherein said processor is executing a plurality of virtual machines, each having a number of processes, and executing a virtual machine monitor.

12. The method of operating said processor system according to claim 11 wherein said disable match indicator is on for all page table entries in said translation buffer for said virtual machine monitor.

16. A processor comprising:

addressing means including a translation buffer for translating virtual addresses to physical addresses, said translation buffer storing a plurality of page table entries, each page table entry containing a page frame number indexed by a virtual address tag;

said translation buffer including means for storing for each said page table entry an address space number, means for storing an address space match entry, and means for storing a match disable indicator; where said address space number corresponds to a process executed on said processor, said match entry is a field having one value indicating that the address space number is to be required to be matched and having another value indicating that the address space number is not required to be matched, and said match disable indicator is an indication having a first value indicating that said match entry is to be ignored and having a second value indicating that said match entry is not to be ignored;

means for maintaining in said processor as part of a state of said processor a current number representing said address space number;

first means for comparing said virtual address tag with a field of a virtual address generated by said processor, and also second means for comparing said address space number with said current number, and

if both of said first and second means for comparing produce a match, and said match entry is of said one value, then said addressing means using said page frame number for a memory reference;

if said means for comparing said virtual address tag with said field of said virtual address produces said match, and if said match entry is of said another value, then said addressing means using said page frame number for said memory reference regardless of whether said second first means for comparing finds that said address space number matches said current number, unless said disable indicator is set to said first value.

18. The processor according to claim 17 including means for generating virtual addresses used for said fetching of instructions and said accessing said external memory for data, said virtual addresses being compared to address tags in said translation buffer.

20. A processor system having a CPU and a memory, said processor system comprising:

a) means in said CPU for fetching instructions from said memory, means in said CPU for decoding said instructions, and means in said CPU for executing said instructions, said means for executing including means for accessing said memory for read and write data;

b) means in said CPU for generating a virtual address used by said means for fetching of instructions and by said means for accessing said memory for data;

c) a page table stored in said memory and containing a plurality of page table entries, each page table entry including a page frame number referencing a different page of said memory;

d) means for translating said virtual address to a physical address for said memory, said means for translating including a translation buffer storing a number of said page table entries;

e) and means for addressing said memory using the page frame number from said translation buffer and using a part of said virtual address;

f) said translation buffer storing for each said page table entry an address tag and an address space number, and storing an address space match entry, where said address space number is a value corresponding to a process executed on said CPU, and said match entry is an indicator having one value indicating that the address space number is to be required to be matched and having another value indicating that the address space number is not required to be matched; said translation buffer also storing for each said page table entry a match disable indicator; means in said CPU for storing a current number representing an address space value maintained as part of a state of said CPU;

g) compare means in said translation buffer for first comparing said address tag with a field of said virtual address generated by said CPU, and also second comparing said address space number with said current number maintained as part of the state of said CPU, and

if both of said first and second comparing by said compare means produce a match, and said match entry is of said one value, then said addressing means using said page frame number for a memory reference; or

if comparing said address tag with said field of said virtual address produces said match, and if said match entry is of said another value, then said addressing means

using said page frame number for said memory reference regardless of whether said address space number matches said current number, unless said disable match indicator is set.

21. The processor system according to claim 20 wherein said CPU is executing a plurality of virtual machines, each having a number of processes, and executing a virtual machine monitor.

22. The processor system according to claim 21 wherein said disable match indicator is set for all page table entries in said translation buffer for said virtual machine monitor.

Refine Search

Search Results -

Term	Documents
(10 AND 7 AND 8).USPT.	1
(L10 AND L7 AND L8).USPT.	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L11

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Friday, January 23, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=ADJ			
L11	L10 and l7 and l8	1	L11
L10	5319760.pn.	1	L10
L9	L8 and l7 and l2	1	L9
L8	device and hardware	139031	L8
L7	virtual$	247654	L7
L6	L5 and l2	1	L6
L5	hardware	177783	L5
L4	L3 and l2	0	L4
L3	harware device	1	L3
L2	6633916.pn.	1	L2
DB=TDBD; PLUR=YES; OP=ADJ			
L1	6633916.pn.	0	L1

h e b b cg b e e ch

END OF SEARCH HISTORY

h e b b cg b e e ch